Online Anomaly Detection Based on Monitoring Traces

Marius Oehler, Alexander Wert, Christoph Heger
{marius.oehler; alexander.wert; christoph.heger}@novatec-gmbh.de
NovaTec Consulting GmbH

Abstract

In modern days, customers expect that Web services work reliably and are available around the clock. A system failure can have a significant negative impact on a company's reputation and economical success. This makes it necessary to continuously monitor software systems in order to detect problems of arising failures early. Existing anomaly detection approaches are taking up this challenge by analyzing aggregated data. Unfortunately, they can suffer from the flaw of averages and are not able to associate an anomaly with individual requests for root cause analysis.

In light of this observation, we introduce an anomaly detection approach that operates on raw and non-aggregated data. This allows us to precisely identify abnormal requests and differentiate them based on various attributes (e.g., business transactions, user location and server). Subsequently, the individual requests can be further analyzed to get a deeper understanding of an anomaly's root cause. We integrated our approach into inspectIT and successfully identified anomalies introduced in an e-commerce reference application.

1 Introduction

These days, computer systems get larger and more dynamic in terms of software and hardware. Especially modern, distributed systems with continuous deployment of changes like patches and new software versions and self-scaling features of the underlying hardware make it impossible to observe and monitor manually these systems.

In order to solve this problem, monitoring systems are used. Such systems are able to combine multiple data streams of various systems. Furthermore, it is possible to aggregate and visualize this data to provide a simplified overview of the whole system and its current state.

However, monitoring systems are not only used to combine, aggregate, persist or visualize data streams but also to provide feedback immediately in case something unexpected happens in the system. Since a huge amount of runtime data is available it can be examined for patterns that refer to existing problems. This can be done automatically and, in addition, notify a responsible person in case of a problem being recognized.

For that reason, it is essential to monitor important software systems in order to get notified immediately when problems arise. Based on this, DevOps teams are able to react promptly to these problems in order to prevent a system failure or to mitigate its impact.

Nevertheless, it is often the case that systems are not monitored due to several reasons. These might be high licensing fees of existing solutions or the effort of setting-up, configuring and maintaining a system is not justified. Another issue is the fact that if an abnormal behavior has been recognized, the root cause of it can often not be determined from the available data because it has been aggregated and the necessary information is unavailable.

In this paper, we solve this problem by introducing an approach that uses monitoring traces for anomaly detection. Furthermore, we integrate additional analysis tools to automatically analyze the data which the anomaly detection feature has considered abnormal to identify its root cause.

For evaluating the elaborated approach, we extend the functionality of an existing open-source tool which is able to collect runtime data of a software system by monitoring and anomaly detection capabilities based on non-aggregated data. This extension allows the monitoring systems to examine the gathered data and precisely determine the root cause of abnormal data.

2 Related Work

There are many publications which address the topic of anomaly detection in software systems. Chandola et al. [5] provide an overview of existing strategies and algorithms that can be used to detect anomalies.

A rule-based detection approach has been published by Chan et al [2]. Here, rules defining the normal behavior are generated based on historical data. These can be used to determine an anomalous behavior.

Song et al. [4] published a model-based approach where the detection process consists of the comparison between the actual system's behavior and a model describing the normal behavior of the system. In this case, the model has to be created manually.

Verbesselt et al. [8] showed that a time series can be decomposed in its individual components. These, in turn, can be expressed as mathematical models. Based on this, a model defining the system's normal

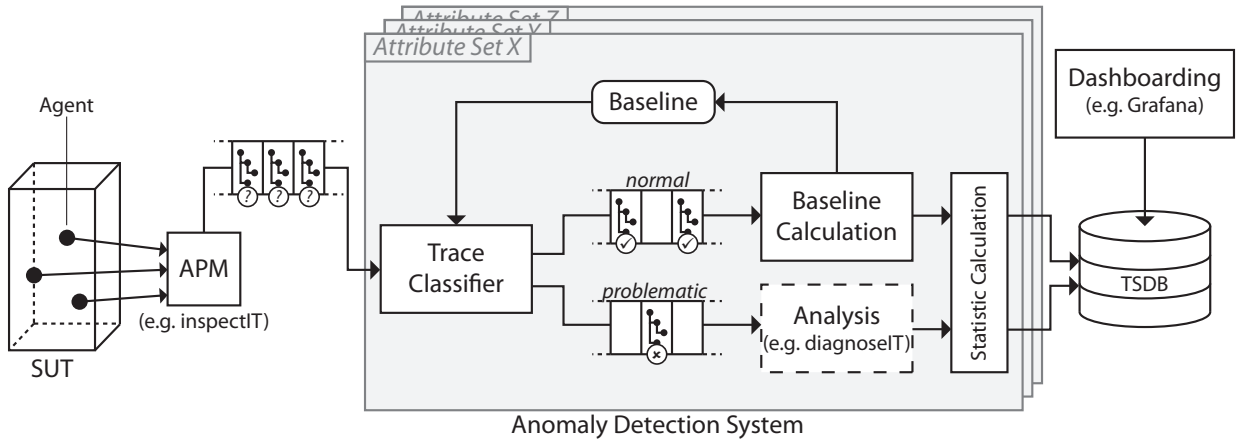

Figure 1: Conceptual architecture of the elaborated Anomaly Detection System.

behavior can be generated automatically. This model can be used to predict the system's behavior. Subsequently, the prediction is compared to the actual behavior. Publications of Kalekar et al. [3], Gelper et al. [6], Goodwin [7], Bielefeld [10] and Frotscher [1, 9] are based on this.

All of the aforementioned approaches are based on a preliminary aggregation of the data, whereas our approach is not limited to this, on the contrary, our approach uses non-aggregated data for the online classification process.

3 Approach

In this section, we introduce our elaborated anomaly detection approach based on the analysis of monitoring traces. These traces can be obtained from, for example, an application performance management (APM) tool, e.g., inspectIT[1]. They have to contain their total duration and, optionally, multiple attributes, like business transaction, geo location, etc.

Figure 1 shows an abstract system architecture of our approach. On the left side of the figure, we have the system under test (SUT), which we monitor. In order to get the necessary runtime data, we have to embed so-called *agents* into the SUT. These agents form a part of the APM tool and are responsible for gathering data. As mentioned before, this data consists of traces which represent, for example, invocation sequences. Once the data is collected, the agent sends it to the APM tool.

In the next step, the monitoring data received by the APM tool is forwarded to the *Anomaly Detection System* (ADS). It represents the elaborated approach. All functionality required for long-term monitoring and anomaly detection is contained in it. The ADS can be a standalone component or embedded into the upstream APM tool.

The data forwarded by the APM tool can be considered a stream of traces. At first, these traces are partitioned according to specific attributes defined by user-defined attribute sets which could consist of business transaction, server location, etc. Each of the resulting groups has its own processing flow. That in turn enables us to use different baselines for different types of traces.

The partitioned stream of traces is going into the *Trace Classifier*. The information contained in the incoming data (e.g., response time) is compared to a derived baseline of this partition. If the baseline is violated, we use an anomaly score to rate the severity of the baseline violation. It is based on the relative distance of the data point to the baseline boundary which can be transformed into an anomaly score using a transfer function. A weighting can be achieved when a logarithm or quadratic function is used. Whenever the severity exceeds a given threshold the trace is classified as *abnormal* and passed to the *Analysis* component which examines the root cause leading to this abnormal trace. At this point, an external analysis application (for example, diagnoseIT [11]) can be used instead of an internal component to examine the trace.

If the severity does not violate the given threshold the trace is considered as *normal* and passed to the *Baseline Calculation* component. We use the double exponential smoothing time series forecasting method in the beginning until enough data is available for Holt-Winters' triple exponential smoothing method. Applying the three-sigma method on the time series forecasting result provides the baseline which is used by the *Trace Classifier* to distinguish the incoming traces between *normal* and *abnormal*. The baseline is computed asynchronously in regular intervals concurrent to the stream processing of the data.

After the *Baseline Calculation* and *Analysis* component, the trace stream flows through the *Statistics Calculation* component. It creates various statistics about the stream, for example, total throughput, ratio of *abnormal* to *normal* traces. Based on these values in combination with predefined thresholds, anomalous behavior of the system's current state can be deduced.

[1] http://www.inspectit.rocks

Additionally, if an anomaly has been detected a responsible person can be notified.

Based on the fact that complex calculations are located in an asynchronous, periodic task, the entire processing flow consists of fast processing steps. Thus, it is possible to scale and parallelize this process and reach a high data throughput without much effort.

Finally, to provide a history and basis for visualization, all data including additional information about whether or not a trace has been considered abnormal is stored in a time series database. On top of this, a visualization framework (e.g., Grafana) can be used to provide a dashboard and to visualize the gathered data.

4 Evaluation

In this section, we present a case study based on *Ticket-Monster*[2] a reference e-commerce Web application. As a base for the implementation of the approach we use the APM tool inspectIT. For the purpose of evaluating the Anomaly Detection System, we injected a performance bottleneck into a business critical transaction which increases its response times with a random delay.

The architecture of the evaluation scenario consists of three servers: a load driver server, an application server including Ticket-Monster and a monitoring server including inspectIT. In this case, inspectIT contains the described *Anomaly Detection System* extension. It analyzes incoming traces and writes the results in an *InfluxDB*[3], the time series database. The data contained in the time series database is visualized by a *Grafana*[4] instance. It is used for visualizing the gathered and analyzed data to provide an overview of the system's current state.

In order to generate load on the Web application, we use the load testing framework *Gatling*[5] which executes a predefined script representing online shopping usage behavior of the Ticket Monster application. Furthermore, one of the virtual users activates the injected performance problem at a random point of time. By doing this, we simulate an anomaly in the system which should be detected by the implemented Anomaly Detection System.

All anomalies caused by the activated problems were promptly detected. Information about the system's state including individual requests could be visualized using the attached dashboarding tool.

5 Conclusion

We presented an approach for online anomaly detection based on non-aggregated data. We implemented the ADS into inspectIT and successfully detected all fault-injected anomalies in the Ticket-Monster Web application. For future work, we plan to evaluate the ADS on a large scale enterprise application in industry and to automatically determine the parameters for partitioning the traces from the incoming monitoring data.

[2] http://developers.redhat.com/ticket-monster/
[3] https://influxdata.com/time-series-platform/influxdb/
[4] http://grafana.org
[5] http://gatling.io

References

[1] Tom Frotscher. "Architecture-based multivariate anomaly detection for software systems". PhD thesis. Kiel University, 1.01.2013.

[2] Philip K Chan, Matthew V Mahoney, and Muhammad H Arshad. "A machine learning approach to anomaly detection". In: *Department of Computer Sciences, Florida Institute of Technology, Melbourne* (2003).

[3] Prajakta S. Kalekar. "Time series forecasting using holt-winters exponential smoothing". In: *Kanwal Rekhi School of Information Technology* (2004).

[4] Xiuyao Song et al. "Conditional anomaly detection". In: *IEEE Transactions on Knowledge and Data Engineering* 19.5 (2007), pp. 631–645.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[6] Sarah Gelper, Roland Fried, and Christophe Croux. "Robust forecasting with exponential and Holt-Winters smoothing". In: *Journal of Forecasting* (2009), n/a–n/a.

[7] Paul Goodwin. "The holt-winters approach to exponential smoothing: 50 years old and going strong". In: *Foresight* 19 (2010), pp. 30–33.

[8] Jan Verbesselt et al. "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series". In: *Remote Sensing of Environment* 114.12 (2010), pp. 2970–2980.

[9] Tom Frotscher. "Prognoseverfahren für das Antwortzeitverhalten von Software-Komponenten". Bachelor Thesis. Christian-Albrechts-Universität zu Kiel, 2011.

[10] Tillmann Carlos Bielefeld. "Online Performance Anomaly Detection for Large-Scale Software Systems". Diploma Thesis. Christian-Albrechts-Universität zu Kiel, 2012.

[11] Christoph Heger et al. "Expert-guided automatic diagnosis of performance problems in enterprise applications". In: (2016).